STAT 401 Chapter 9.1–9.3, 9.6, 10.1

Zepu Zhang
November 23, 2010

(This note is much simplified compared to the book. This chapter of the book serves only to help you understand this note.)

1 Model selection

Model "selection" in this chapter means how to determine the form of the regression model, including what predictors to include, and the functional form of each predictor. The specific form of each predictor is determined by subject-matter knowledge and complexity of the response curve (nonlinear? polynomial? log? ...). Here we assume the forms have been chosen, and in total there are P candidate predictors: X_0, X_1, ..., X_{P-1}. The task of model selection is to determine what of these are eventually included in the regression model.

We first need to choose a criterion for comparing the "quality" of two models (that is, two different choices of predictors). Such a criterion centers on two things:

1. fitting to data: the closer the better;
2. number of predictors: the fewer the better.

Next we want a somewhat automatic algorithm to search for the "best" set of predictors to form the ultimate model. Such routines exist. However, there is no worry-free recipe for this task. Examination by the human eye and subjective judgement is unavoidable. Out of this consideration, we are not going to learn those "automatic" procedures.

We will learn about one of the most commonly used model selection criterion: AIC.

We'll make a distinction between the number of coefficients (β's) and the number of predictors (or "terms")—for a qualitative predictor, it corresponds to more than one β. Adding or dropping a qualitative predictor changes the number of parameters in the model (df of the model) by more than 1 at once.

Akaike's information criterion: AIC

This criterion of model quality, like alternatives, is based on SSE or SSR, with penalty for the number of parameters.

$$\text{AIC}_p = n \log(\text{SSE}_p/n) + 2p$$

n is fixed; SSE_p is the SSE of the (reduced) model with p parameters.

A good model has small SSE. If we introduce more predictors, SSE will decrease. If the decrease in SSE is offset by the increase in p, AIC could go up, suggesting the model is getting worse. This is how the AIC criterion penalizes model complexity (i.e., more parameters).

A smaller AIC suggests a better model. Bearing in mind that subjective judgment is always needed, a simplistic description of model selection is: find the combination of predictors that has the smallest AIC.

If at some point we have a model with certain predictors. There are several additional predictors to be considered for inclusion. We fit the model that has one more predictor (using one of the candidate predictors at a time), identify the new model with the smallest AIC. If this new AIC is smaller than the AIC of the current model, then adopt this new model. The newly included predictor is usually the one with the largest extra SSR.

```
> data <- read.table('RealEstateSales.txt', header = TRUE)
> data <- data[data$Style %in% c(1,2,3,7), ]
> data$Style <- factor(LETTERS[data$Style])
> data$Price <- data$Price / 100000
>     # Change unit of price so that the numbers are not so big.
>
> print(names(data))
 [1] "ID"         "Price"      "SquareFeet" "NumBed"     "NumBath"
 [6] "AC"         "GarageSize" "Pool"       "Year"       "Quality"
[11] "Style"      "LotSize"    "Highway"
>
> data <- data[, c('Price', 'SquareFeet', 'Style',
+             'NumBed', 'NumBath', 'GarageSize', 'AC')]
>     # 'Style': qualitative
>     # 'AC': 0 or 1
>     # 'GarageSize': number of car spots, integer.
>
> myfit <- lm(Price ~ ., data)
> print(myfit)
```

```
Call:
lm(formula = Price ~ ., data = data)

Coefficients:
(Intercept)   SquareFeet       StyleB       StyleC       StyleG       NumBed
  -1.446231     0.001578    -0.279666    -0.222118    -0.816444    -0.126559
    NumBath   GarageSize           AC
   0.234802     0.324000     0.140132


> #################
> #### Block 1 ####
> #################
```

The functions `anova` and `aov` provide extra SSR of the predictors as they are added into the model one by one, i.e. sequentially. (See a previous lecture note.)

Because the extra SSR depends on what predictors are already in the model, the result depends on the order in which the predictors are added into the model.

Note the `Df` in the output. It is the number of parameters the new predictor introduces to the model. Note the `Df` of `Style` in particular.

```
>
> print(anova(myfit))
Analysis of Variance Table

Response: Price
            Df Sum Sq Mean Sq   F value    Pr(>F)
SquareFeet   1 625.30  625.30 1248.1854 < 2.2e-16 ***
Style        3  34.73   11.58   23.1096 5.892e-14 ***
NumBed       1   1.10    1.10    2.1887    0.1397
NumBath      1  15.39   15.39   30.7268 4.996e-08 ***
GarageSize   1  15.86   15.86   31.6560 3.191e-08 ***
AC           1   1.00    1.00    2.0038    0.1576
Residuals  463 231.95    0.50
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
>
> # 'aov' provides the same info in a more concise form.
> # Compare the numbers below with the output of 'anova'.
> print(aov(myfit))
Call:
   aov(formula = myfit)
```

```
Terms:
                SquareFeet    Style   NumBed  NumBath GarageSize       AC
Sum of Squares    625.3024  34.7316   1.0965  15.3932    15.8587   1.0038
Deg. of Freedom          1        3        1        1          1        1
                Residuals
Sum of Squares   231.9487
Deg. of Freedom       463

Residual standard error: 0.7077917
Estimated effects may be unbalanced
>
> # For 'aov', you can provide the formula directly
> # instead of the fitted object:
> print(aov(Price ~ ., data))
Call:
   aov(formula = Price ~ ., data = data)

Terms:
                SquareFeet    Style   NumBed  NumBath GarageSize       AC
Sum of Squares    625.3024  34.7316   1.0965  15.3932    15.8587   1.0038
Deg. of Freedom          1        3        1        1          1        1
                Residuals
Sum of Squares   231.9487
Deg. of Freedom       463

Residual standard error: 0.7077917
Estimated effects may be unbalanced
>
> # You can not do this with 'anova'.
> # 'anova' needs a fitted model object as its first argument.
```

The result of 'anova' and 'aov' depends on the order in which the predictors enter the model. Note the order of the predictors above. How did R choose this order?

```
> print(names(data))
[1] "Price"      "SquareFeet" "Style"      "NumBed"     "NumBath"
[6] "GarageSize" "AC"
```

This shows the order of predictors is simply the order in which they appear in the 'data' data.frame. R did not do any alphabetic ordering etc.

If we change the order of the predictors, the result is different:

```
> print(aov(lm(Price ~ AC + SquareFeet + NumBath
+     + GarageSize + NumBed + Style, data)))
```

```
Call:
   aov(formula = lm(Price ~ AC + SquareFeet + NumBath + GarageSize +
    NumBed + Style, data))

Terms:
                       AC SquareFeet  NumBath GarageSize   NumBed    Style
Sum of Squares   66.6647   562.2019   5.5381    20.3213   6.7139  31.9462
Deg. of Freedom       1          1        1          1        1        3
                Residuals
Sum of Squares   231.9487
Deg. of Freedom       463

Residual standard error: 0.7077917
Estimated effects may be unbalanced
>


>
> ##################
> #### Block 2 ####
> ##################
```

The function `drop1` takes a fitted model, fits a reduced model with one of the predictors dropped, and shows the df of the dropped term, the decrease in SSR of the model (compared with the original model), the SSE of the reduced model, and the AIC of the reduced model.

```
> print(drop1(myfit))
Single term deletions

Model:
Price ~ SquareFeet + Style + NumBed + NumBath + GarageSize +
    AC
           Df Sum of Sq    RSS     AIC
<none>                  231.95 -317.34
SquareFeet  1   174.049 406.00  -55.10
Style       3    31.946 263.89 -262.43
NumBed      1     4.754 236.70 -309.76
NumBath     1    10.734 242.68 -297.99
GarageSize  1    13.831 245.78 -292.00
AC          1     1.004 232.95 -317.30
```

To understand what these numbers are, let's verify several of them.

The first row contains the SSE (Residual Sum of Squares, 'RSS') and AIC of the full model:

```
> print(deviance(myfit))
[1] 231.9487
>
> n <- nrow(data)
> print(n * log(deviance(myfit) / n) + 2 * length(coef(myfit)))
[1] -317.3384
>
> # Each of the other rows drops by term from the full model,
> # and shows the extra SSR of the dropped term,
> # the SSE ('RSS' in R output) of the reduced model,
> # and AIC of the reduced model.
>
> # Note the Df of each dropped term.
> # In particular, the 'Df' of 'Style'.
>
> newfit <- update(myfit, . ~ . - SquareFeet)
> print(deviance(newfit) - deviance(myfit))
[1] 174.0488
> print(deviance(newfit))
[1] 405.9975
> print(n * log(deviance(newfit) / n) + 2 * length(coef(newfit)))
[1] -55.09831
>
> newfit <- update(myfit, . ~ . - NumBed)
> print(deviance(newfit) - deviance(myfit))
[1] 4.753524
> print(deviance(newfit))
[1] 236.7022
> print(n * log(deviance(newfit) / n) + 2 * length(coef(newfit)))
[1] -309.7631
```

Exercise If you are to drop one term from the model, what term will you drop? Does the change make the model better or worse by the AIC criterion?

Although rarely necessary, we could specify what terms to consider dropping by the 'scope' argument. Without this, each term will be dropped in turn.

```
> print(drop1(myfit, scope = ~ SquareFeet + Style + NumBed))
Single term deletions

Model:
Price ~ SquareFeet + Style + NumBed + NumBath + GarageSize +
    AC
           Df Sum of Sq    RSS     AIC
<none>                  231.95 -317.34
SquareFeet  1   174.049 406.00  -55.10
```

```
Style       3    31.946 263.89 -262.43
NumBed      1     4.754 236.70 -309.76
```

There is an analogous function `add1`, which begins with a simpler model, e.g.
`lm(Price ~ 1, data)`
and considers adding each candidate term specified by `scope`.

For some technical reason, we prefer `drop1` to `add1`, that is, we recommend starting with a full model and examine what terms to drop.

```
> m1 <- lm(Price ~ 1 + SquareFeet, data)
> print(add1(m1, scope = ~ . + NumBed + Style + AC))
Single term additions

Model:
Price ~ 1 + SquareFeet
       Df Sum of Sq    RSS     AIC
<none>              300.03 -209.86
NumBed  1     2.525 297.51 -211.85
Style   3    34.732 265.30 -261.93
AC      1     3.564 296.47 -213.50
>     # Note the 'scope' argument.
>     # On the RHS of '~' we use '.' to mean what's already
>     # in the model.
>     # To the left of '~' you can write '.' or omit it.
```

Added-variable plots

(Self reading.)

To examine the marginal importance of X_k while the other Xs are also in the model, plot the residuals of Y modeled by the other X's against the residuals of X_k modeled by the other X's.

This residual-residual plot

1. Shows the potential of X_k in help modeling Y.
2. May suggest the functional form of X_k when it is added to the model for Y.

Example Figure 10.1, page 385.

2 Model validation

Model "validation" means to check how the regression model, estimated based on a dataset, works for a new dataset. This is usually measured by how well the model "predicts" the new data (Y). The "new" data can be either

- genuinely new data obtained after the model was built, or
- part of the original data that are kept aside and not used at the model estimation stage.

Suppose the model has been estimated with some data. Examine the predictive capability of the model with a new dataset:

$$\text{MSPR} = \frac{\sum_{i=1}^{n^*}(Y_i - \hat{Y}_i)^2}{n^*}$$

where n^* is the number of new data, and MSPR stands for "mean squared prediction error".

If MSPR is "close" to the MSE in the model-building step, it suggests the model performs consistently with both the model-building data and the new data. Otherwise (i.e. MSPR is much larger), the model can not be extended to the new data. If one must use this model, then MSPR rather than MSE should be taken as a measure of the model's predictive error.

Note 1. The original data may be split into a training (or model-building) set and a validation (or testing) set. This is known as cross-validation, and is a general idea.

2. If the size of the data set is not that large, make sure the training set is large enough so that the model estimation is reasonably reliable.

3. We do not have to be stuck with one particular split. Actually we can split the data randomly and do the model-building and validation. Split it again (randomly and differently), repeat. Average the finding in some way. This is the general idea of "cross-validation".

4. After it is decided the model is good for production use, use all the data (no split) to obtain a final (and better) estimation of the model parameters.